

BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	


```
1 0001 0 MODULE writeboot ( ! Writes boot block code and data into LBN 0
2 0002 0 IDENT = 'V04-000',
3 0003 0 MAIN = write_boot
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1 FACILITY:
33 0033 1
34 0034 1 WRITEBOOT
35 0035 1
36 0036 1 ABSTRACT:
37 0037 1
38 0038 1 The purpose of this utility is to write a BOOTable program into
39 0039 1 LBN 0 of a system disk or TU58. This BOOTable program will
40 0040 1 contain within its first three longwords, the starting LBN and
41 0041 1 size of a primary VMS bootstrap file located on this same system
42 0042 1 disk or TU58 and also the relative location in memory where the
43 0043 1 primary bootstrap should be loaded. The system disk or TU58 may be
44 0044 1 a FILES11 (ODS-2) or an RT-11 formatted device.
45 0045 1
46 0046 1 ENVIRONMENT:
47 0047 1
48 0048 1 VAX/VMS operating system, requires LOG_IO privilege. Assumes
49 0049 1 bootstrap file is VMB.EXE unless otherwise specified.
50 0050 1
51 0051 1 AUTHOR:
52 0052 1
53 0053 1 Carol Peters 20 June 1979
54 0054 1
55 0055 1 REVISION HISTORY:
56 0056 1
57 0057 1 V03-003 TCM0001 Trudy C. Matthews 10-Aug-1983
```

:	58	0058	1	:	Ensure that the bootfile is contiguous before writing the
:	59	0059	1	:	boot block.
:	60	0060	1	:	
:	61	0061	1	:	V03-002 RAS0175 Ron Schaefer 28-Jul-1983
:	62	0062	1	:	Eliminate useless reference to FAB\$V_UFM.
:	63	0063	1	:	
:	64	0064	1	:	V03-001 PCA1006 Paul C. Anagnostopoulos 9-Dec-1982
:	65	0065	1	:	Modify the initialization of the BOOTBLOCK.EXE RMS control
:	66	0066	1	:	blocks so that the logical name BOOTBLOCK can be used.
:	67	0067	1	:	On systems with library disks, this image is not on
:	68	0068	1	:	the system disk.
:	69	0069	1	:	
:	70	0070	1	:	V02-002 STJ0054 Steven T. Jeffreys, 29-Jun-1981
:	71	0071	1	:	Changed external routine references to use general addressing mode.
:	72	0072	1	:	
:	73	0073	1	:	Robert Rappaport 10 Aug 1979
:	74	0074	1	:	Major changes to accomodate RT-11 format devices.
:	75	0075	1	:	
:	76	0076	1	:	Steve Jeffreys 12 Nov 1979
:	77	0077	1	:	- Enable WRITEBOOT to accept input from an indirect command file.
:	78	0078	1	:	- Remove verify prompt for target device.
:	79	0079	1	:	- Add prompt for VBN of code in boot file.
:	80	0080	1	:	- Allow boot code to be loaded at an arbitrary address.
:	81	0081	1	:	--


```
: 140      0139 1      vbn_descrip      : BLOCK [8, BYTE] INITIAL      ! Prompt string for VBN
: 141      0140 1
: 142      0141 1      (REP 3 OF (0),
: 143      0142 1      dsc$$_class_d,
: 144      0143 1      REP 4 OF (0))),
: 145      0144 1
: 146      0145 1      priboo_fab      : $FAB_DECL,      ! Primary bootstrap file's FAB.
: 147      0146 1      bootbl_fab     : $FAB_DECL,      ! Boot block file's FAB.
: 148      0147 1
: 149      0148 1      priboo_filnam   : VECTOR [nam$_maxrss, BYTE], ! Primary bootstrap file name after open.
: 150      0149 1      priboo_exp_name : VECTOR [nam$_maxrss, BYTE], ! Primary bootstrap file name before open.
: 151      0150 1
: 152      0151 1 !      bootbl_filnam   : VECTOR [nam$_maxrss, BYTE], ! Boot block file name after open.
: 153      0152 1 !      bootbl_exp_name : VECTOR [nam$_maxrss, BYTE], ! Boot block file name before open.
: 154      0153 1
: 155      0154 1 !      result_nam_blk  : $NAM (RSA = priboo_filnam), ! Related file NAM block.
: 156      0155 1
: 157      0156 1 !      bootbl_nam_blk  : $NAM (      ! Name block for BOOTBLOCK.EXE.
: 158      0157 1 !      RSA = bootbl_filnam,
: 159      0158 1 !      RSS = nam$_maxrss,
: 160      0159 1 !      ESA = bootbl_exp_name,
: 161      0160 1 !      ESS = nam$_maxrss,
: 162      0161 1 !      RLF = result_nam_blk),
: 163      0162 1
: 164      P 0163 1      priboo_nam_blk : $NAM (      ! Name block for primary bootstrap.
: 165      P 0164 1      RSA = priboo_filnam,
: 166      P 0165 1      RSS = nam$_maxrss,
: 167      P 0166 1      ESA = priboo_exp_name,
: 168      0167 1      ESS = nam$_maxrss),
: 169      0168 1
: 170      0169 1      priboo_xabfhc   : $XABFHC (),      ! Primary bootstrap file header characteristics bloc
: 171      0170 1 !      bootbl_xabfhc   : $XABFHC (),      ! Boot block file header characteristics block.
: 172      0171 1
: 173      0172 1      privilege_mask : BLOCK [8, BYTE],
: 174      0173 1      getjpi_itemlist : BLOCK [4, LONG] INITIAL
: 175      0174 1      (WORD (8, JPI$_PROCPRIV),
: 176      0175 1      LONG (privilege_mask, 0,0)),
: 177      0176 1      io_stat_block   : VECTOR [2],
: 178      0177 1
: 179      0178 1      two_block_buf   : BLOCK [1024, BYTE],
: 180      0179 1
: 181      0180 1      bootdev_descrip : BLOCK [8, BYTE],
: 182      0181 1      bootdev_chan   : WORD,
: 183      0182 1      load_adr      : LONG,
: 184      0183 1      devchar_buff   : BLOCK [3, LONG] ! Buffer to receive device characteristics.
: 185      0184 1      INITIAL (LONG (REP 3 OF (0))),
: 186      0185 1
: 187      0186 1
: 188      0187 1      devchar_descrip : BLOCK [8, BYTE] ! Descriptor for Device characteristics.
: 189      0188 1      INITIAL (LONG (12, devchar_buff)),
: 190      0189 1
: 191      0190 1      filnam_descrip  : BLOCK [8, BYTE] ! Descriptor for just the file name (no device or di
: 192      0191 1      INITIAL (BYTE (REP 8 OF (0))),
: 193      0192 1      filspec_descrip : BLOCK [8, BYTE] ! Descriptor for file spec returned from $PARSE.
: 194      0193 1      INITIAL (LONG (0, priboo_exp_name)),
: 195      0194 1
: 196      0195 1      vbn      : VECTOR [1, LONG],      ! VBN of boot file code
```



```
: 197      0196 1      stat_block      : VECTOR [2, LONG];          ! Area to hold LBN and size of specified file.
: 198      0197 1
: 199      0198 1 BIND
: 200      0199 1      block_buffer = two block buf : BLOCK [512, BYTE],
: 201      0200 1      logio_msg = UPLIT BYTE (%ASCII 'You lack LOG IO privilege.') : VECTOR [, LONG],
: 202      0201 1      vbn_bnds_msg = UPLIT BYTE (%ASCII 'VBN must be >= 1.') : VECTOR [, LONG],
: 203      0202 1      notcontig_msg = UPLIT BYTE (%ASCII 'Boot file is not contiguous.')
: 204      0203 1      : VECTOR [, LONG],
: 205      0204 1      remount_msg = UPLIT BYTE (%ASCII 'You lack READ and/or WRITE access to TARGET DEVICE. DISMOUNT and
: 206      0205 1      : VECTOR [, LONG],
: 207      0206 1      ascii_bracket = UPLIT BYTE (%ASCII ']') : VECTOR [, LONG],
: 208      0207 1      prompt_buffer = UPLIT BYTE (%ASCII 'Target system device (and boot file if not VMB.EXE): ') : VECTOR
: 209      0208 1      prompt2_buffer = UPLIT BYTE (%ASCII 'Enter load address of primary bootstrap in HEX (default is 200)
: 210      0209 1      prompt3_buffer = UPLIT BYTE (%ASCII 'Enter VBN of boot file code (default is 1) : ') : VECTOR [, LON
: 211      0210 1      priboo_def_name = UPLIT BYTE (%ASCII '[SYSEXE]VMB.EXE') : VECTOR [, LONG];
: 212      0211 1
: 213      0212 1 LITERAL
: 214      0213 1      dev_offset      = 18,
: 215      0214 1      logio_length    = 26,
: 216      0215 1      vbn_bnds_len    = 17,
: 217      0216 1      remount_length  = 77,
: 218      0217 1      notcontig_length = 28,
: 219      0218 1      prompt_length   = 53,          ! Length of prompt.
: 220      0219 1      prompt2_length  = 65,          ! Length of prompt2.
: 221      0220 1      prompt3_length  = 45,          ! Length of prompt3.
: 222      0221 1      bootname_length = 15;          ! Length of VMB.EXE.
: 223      0222 1
: 224      0223 1 OWN
: 225      0224 1      yes_no_buf      : BLOCK [1, BYTE] INITIAL (BYTE (0)),
: 226      0225 1      yes_no_descrip  : BLOCK [8, BYTE]
: 227      0226 1      INITIAL (LONG (1, yes_no_buf)),
: 228      0227 1
: 229      0228 1      logio_descrip   : BLOCK [8, BYTE]
: 230      0229 1      INITIAL (LONG (logio_length, logio_msg)),
: 231      0230 1      vbn_bnds_descrip: BLOCK [8, BYTE]
: 232      0231 1      INITIAL (LONG (vbn_bnds_len, vbn_bnds_msg)),
: 233      0232 1      remount_descrip : BLOCK [8, BYTE]
: 234      0233 1      INITIAL (LONG (remount_length, remount_msg)),
: 235      0234 1      notcontig_descrip : BLOCK [8, BYTE]
: 236      0235 1      INITIAL (LONG (notcontig_length, notcontig_msg)),
: 237      0236 1      bracket_descrip : BLOCK [8, BYTE]
: 238      0237 1      INITIAL (LONG (1, ascii_bracket)); ! Descriptor for constant string consisting of ']'.
: 239      0238 1
: 240      0239 1
```

```
242 0240 1 ROUTINE write_boot = ! Writes the boot block.
243 0241 1
244 0242 1
245 0243 1 Functional description:
246 0244 1
247 0245 1 1. Prompts for target system device and optional boot file spec.
248 0246 1
249 0247 1 2. Determines if target device is Files-11 or FOREIGN.
250 0248 1
251 0249 1 3. Determines starting LBN and size of VMB.EXE (or specified file)
252 0250 1 on the target system device specified by the user in step #1.
253 0251 1 In case of Files-11 device this means opening file with
254 0252 1 an XABFHC specified in the FAB.
255 0253 1 In case of a FOREIGN device this means calling external
256 0254 1 routine 'RTFSOPENFILE'.
257 0255 1
258 0256 1 - Prompt for VBN of boot file code.
259 0257 1
260 0258 1 4. Prompts for memory location where primary bootstrap should be
261 0259 1 loaded in memory.
262 0260 1
263 0261 1 5. Opens SYS$SYSTEM:BOOTBLOCK.EXE on the current system disk.
264 0262 1
265 0263 1 6. Reads VBN #0 of SYS$SYSTEM:BOOTBLOCK.EXE into buffer.
266 0264 1
267 0265 1 7. Modifies buffer by placing starting LBN, size and memory location
268 0266 1 obtained in steps #3 and #4 above, into the buffer at the
269 0267 1 appropriate places.
270 0268 1
271 0269 1 8. Writes buffer containing modified copy of SYS$SYSTEM:BOOTBLOCK.EXE
272 0270 1 into LBN #0 of target system device specified by user in step #1.
273 0271 1
274 0272 1 9. Closes files.
275 0273 1
276 0274 1 Inputs:
277 0275 1
278 0276 1 none
279 0277 1
280 0278 1 Outputs:
281 0279 1
282 0280 1 R0 contains a status code.
283 0281 1
284 0282 1 --
285 0283 1
286 0284 2 BEGIN
287 0285 2
288 0286 2 LOCAL
289 0287 2 index,
290 0288 2 status;
291 0289 2
292 0290 2 ! Issue a $GETJPI system service call to discover whether the process
293 0291 2 executing WRITEBOOT has LOG_IO privilege. If not, don't allow process
294 0292 2 to write on the target system disk.
295 0293 2
296 0294 2
297 0295 3 IF NOT (status = $getjpi (efn = 3, itmlst = getjpi_itemlist, iosb = io_stat_block))
298 0296 2 THEN RETURN .status;
```



```
299 0297 2 IF NOT .privilege_mask [prv$v_log_io]
300 0298 2 THEN BEGIN
301 0299 2     lib$put_output (logio_descrip);
302 0300 2     RETURN SS$_NOPRIV;
303 0301 2     END;
304 0302 2
305 0303 2
306 0304 2     Prompt for the target system device name optionally followed by the
307 0305 2     name of a primary bootstrap file.
308 0306 2
309 0307 2
310 0308 2     prompt_descrip [dsc$w_length] = prompt_length;
311 0309 2     prompt_descrip [dsc$a_pointer] = prompt_buffer;
312 0310 2
313 0311 2     IF .priboo_descrip [dsc$w_length] NEQ 0
314 0312 2         THEN lib$free1_d0 (priboo_descrip);      ! deallocate previous string.
315 0313 2
316 0314 2     WHILE .priboo_descrip [dsc$w_length] EQL 0
317 0315 2     DO
318 0316 2         BEGIN
319 0317 2             status = lib$get_input (priboo_descrip, prompt_descrip);
320 0318 2             IF NOT .status
321 0319 2                 THEN RETURN .status;
322 0320 2             END;
323 0321 2
324 0322 2
325 0323 2     Translate all lower case alphabetic characters to upper case so that
326 0324 2     an RMS translation will work.
327 0325 2
328 0326 2
329 0327 2     INCR count FROM 0 TO (.priboo_descrip [dsc$w_length] - 1)
330 0328 2     DO
331 0329 2         BEGIN
332 0330 2         BIND
333 0331 2             file_spec = .priboo_descrip [dsc$a_pointer] : VECTOR [, BYTE];
334 0332 2             IF ((.file_spec [.count] GEQ 'a') AND (.file_spec [.count] LEQ 'z'))
335 0333 2                 THEN file_spec [.count] = .file_spec [.count] - %x'20';
336 0334 2             END;
337 0335 2
338 0336 2
339 0337 2     Determine if the target device is Files-11 or FOREIGN. Do this
340 0338 2     by $PARSEing the given file spec using the default of [SYSEXE]VMB.EXE
341 0339 2     and by specifying a NAM block. With NAM block we obtain the device
342 0340 2     name in the nam$dvi field and we build a string descriptor for this
343 0341 2     string and use system service $GETDEV to get the device characteristics.
344 0342 2
345 0343 2
346 0344 2
347 P 0345 2 $FAB_INIT (
348 P 0346 2     FAB = priboo_fab,
349 P 0347 2     FAC = <GET>,
350 P 0348 2     FNA = .priboo_descrip [dsc$a_pointer],
351 P 0349 2     FNS = .priboo_descrip [dsc$w_length],
352 P 0350 2     DNA = priboo_def_name,
353 P 0351 2     DNS = bootname_length,
354 P 0352 2     FOP = <NAM>,
355 P 0353 2     NAM = priboo_nam_blk,
```

```
356      XAB = priboo_xabfhc);
357      IF NOT (status = $PARSE (FAB = priboo_fab))
358      THEN RETURN .status;
359
360      bootdev_descrip[dsc$w_length] = .(priboo_nam_blk[nam$st_dvi]) <0,8>;
361      bootdev_descrip[dsc$a_pointer] = priboo_nam_blk[nam$st_dvi] + 1;
362
363      P 0361
364      IF NOT (status = $GETDEV (DEVNAM = bootdev_descrip,
365      PRIBUF = devchar_descrip))
366      THEN RETURN .status;
367
368      :
369      :
370      : At this point we have the target device characteristics. If the
371      : device is FOREIGN then we isolate the file name in the expanded
372      : file spec and build a string descriptor for this substring.
373      : Next we call RTF$TARGET_DEV to record the name of the target device.
374      : Then we call RTF$OPENFILE to get the starting LBN and size. If
375      : on the other hand the device is Files-11, then we simply open the file.
376      : The purpose of the open is to load the size and starting LBN of the
377      : file into the XABFHC block produced by RMS. In this latter case of a
378      : Files-11 device we then copy this data out of the XABFHC block into
379      : the OWN variable stat_block.
380      :
381      :
382      IF .devchar_buff[dev$w_for] ! i.e. if FOREIGN
383      THEN BEGIN
384          filspec_descrip[dsc$w_length] = .priboo_nam_blk[nam$b_esl];
385          index = lib$index (filspec_descrip, bracket_descrip);
386          filnam_descrip[dsc$a_pointer] = .filspec_descrip[dsc$a_pointer] + .index;
387          filnam_descrip[dsc$w_length] = .filspec_descrip[dsc$w_length] - .index;
388
389          RTF$TARGET_DEV (bcotdev_descrip);
390
391          IF NOT (status = RTF$OPENFILE (filnam_descrip,
392          two_block_buf,
393          stat_block))
394          THEN BEGIN
395              lib$put_output (remount_descrip);
396              RETURN .status;
397          END;
398      ELSE BEGIN
399          IF NOT (status = $RMS_OPEN (FAB = priboo_fab))
400          THEN RETURN .status;
401
402          stat_block[0] = .priboo_xabfhc[xab$l_sbn];
403          IF .priboo_xabfhc[xab$l_sbn] EQL 0
404          THEN BEGIN
405              lib$put_output (notcontig_descrip);
406              $RMS_CLOSE (FAB = priboo_fab);
407              RETURN $$$_FILNOTCNTG;
408          END;
409          IF .priboo_xabfhc[xab$w_ffb] NEQ 0
410          THEN stat_block[1] = .priboo_xabfhc[xab$l_ebk]
411          ELSE stat_block[1] = .priboo_xabfhc[xab$l_ebk] - 1;
412
```



```
413 0411 3      $RMS_CLOSE (FAB = priboo_fab);
414 0412      END;
415 0413
416 0414
417 0415
418 0416      Prompt the user for the VBN of the boot file code.
419 0417
420 0418
421 0419      prompt_descrip[dsc$w_length] = prompt3_length; ! Set up prompt descriptor
422 0420      prompt_descrip[dsc$a_pointer] = prompt3_buffer;
423 0421
424 0422      status = 0;
425 0423      WHILE NOT .status
426 0424      DO
427 0425          BEGIN
428 0426              IF .vbn_descrip[dsc$w_length] NEQ 0
429 0427                  THEN lib$free1_d0 (vbn_descrip);          ! Deallocate previous string
430 0428
431 0429              IF NOT (status = lib$get_input (vbn_descrip, prompt_descrip)) ! Prompt for VBN
432 0430                  THEN RETURN .status;
433 0431
434 0432              IF .vbn_descrip[dsc$w_length] NEQ 0          ! Convert string to decimal #
435 0433                  THEN status = ots$cvt_tz_l (vbn_descrip,vbn)
436 0434                  ELSE vbn = 1;          ! Default VBN
437 0435
438 0436              IF .vbn LSS 1          ! Check for VBN < 1
439 0437                  THEN
440 0438                  BEGIN
441 0439                      IF NOT (status = lib$put_output (vbn_bnds_descrip))
442 0440                          THEN RETURN .status;
443 0441                      status = 0;
444 0442                  END;
445 0443              END;          ! End of VBN prompt WHILE loop
446 0444
447 0445      stat_block[0] = .stat_block[0] + (.vbn - 1);          ! Update LBN to point to boot code
448 0446
449 0447
450 0448      Open the bootblock file (called SYSS$SYSTEM:BOOTBLOCK.EXE) located on the
451 0449      system disk. Ensure that the logical name BOOTBLOCK will work.
452 0450
453 0451
454 0452      $FAB_INIT (
455 0453          FAB = bootbl_fab,
456 0454          DNM = 'SYSS$SYSTEM:.EXE',
457 0455          FAC = <BIO>,
458 0456          FNM = 'BOOTBLOCK',
459 0457          FOP = <UFO>;
460 0458      IF NOT (status = $RMS_OPEN (FAB = bootbl_fab))
461 0459          THEN RETURN .status;
462 0460
463 0461
464 0462
465 0463      Read the first block of BOOTBLOCK.EXE into a page-long buffer in
466 0464      memory.
467 0465
468 0466
469 0467      IF NOT (status = $qiow (
```

```

: 470 P 0468      CHAN = ,bootbl_fab [fab$l_stv],
: 471 P 0469      FUNC = ios_readvblk,
: 472 P 0470      P1 = block_buffer,
: 473 P 0471      P2 = 512,
: 474 P 0472      P3 = 1))
: 475 P 0473      THEN RETURN .status;
: 476 P 0474
: 477 P 0475      !
: 478 P 0476      ! Here we prompt the user for the relative memory location that he wants
: 479 P 0477      ! the primary bootstrap loaded into.
: 480 P 0478
: 481 P 0479
: 482 P 0480      prompt_descrip[dsc$w_length] = prompt2_length;
: 483 P 0481      prompt_descrip[dsc$a_pointer] = prompt2_buffer;
: 484 P 0482
: 485 P 0483      status = 0;      ! Set to false for following loop.
: 486 P 0484
: 487 P 0485      WHILE NOT .status
: 488 P 0486      DO
: 489 P 0487      BEGIN
: 490 P 0488          IF .loadadr_descrip[dsc$w_length] NEQ 0
: 491 P 0489              THEN lib$free1_dd (loadadr_descrip);
: 492 P 0490
: 493 P 0491          status = lib$get input (loadadr_descrip, prompt_descrip);
: 494 P 0492          IF NOT .status THEN RETURN .status;
: 495 P 0493
: 496 P 0494          IF .loadadr_descrip[dsc$w_length] NEQ 0
: 497 P 0495              THEN status = ots$cvt_tz_l(loadadr_descrip, load_adr)
: 498 P 0496              ELSE load_adr = 512;      ! Default
: 499 P 0497      END;
: 500 P 0498
: 501 P 0499      !
: 502 P 0500      ! Load the starting LBN, size and relative load location into the first
: 503 P 0501      ! 3 longwords of the buffer containing the BOOTBLOCK code.
: 504 P 0502
: 505 P 0503
: 506 P 0504      block_buffer [bbl_l_filesize] = .stat_block[1];      ! Copy filesize.
: 507 P 0505      block_buffer [bbl_w_hiordlbn] = .(stat_block[0])<16,16>; ! Swap LBN words for
: 508 P 0506      block_buffer [bbl_w_loordlbn] = .(stat_block[0])<0,16>; ! DSC
: 509 P 0507      block_buffer [bbl_l_loadadr] = .load_adr;      ! Copy where to load
: 510 P 0508      ! primary bootstrap
: 511 P 0509
: 512 P 0510
: 513 P 0511      !
: 514 P 0512      ! Assign a channel to target device.
: 515 P 0513
: 516 P 0514
: 517 P 0515      IF NOT (status = $assign (
: 518 P 0516          DEVNAM = bootdev_descrip,
: 519 P 0517          CHAN = bootdev_chan))
: 520 P 0518      THEN RETURN .status;
: 521 P 0519
: 522 P 0520      !
: 523 P 0521      ! Copy the page-long buffer into LBN 0 of the target system device.
: 524 P 0522
: 525 P 0523
: 526 P 0524      IF NOT (status = $qiow (
```



```
.. 527      P 0525      CHAN = .bootdev.chan,  
.. 528      P 0526      FUNC = io$_writeblk,  
.. 529      P 0527      P1 = block_buffer,  
.. 530      P 0528      P2 = 512,  
.. 531      P 0529      P3 = 0))  
.. 532      P 0530      THEN RETURN .status;  
.. 533      P 0531      :  
.. 534      P 0532      : Close the open files.  
.. 535      P 0533      :  
.. 536      P 0534      :  
.. 537      P 0535      $RMS_CLOSE (FAB = bootbl_fab);  
.. 538      P 0536      :  
.. 539      P 0537      : Return with success status.  
.. 540      P 0538      :  
.. 541      P 0539      :  
.. 542      P 0540      :  
.. 543      P 0541      :  
.. 544      P 0542      RETURN S$$_NORMAL;  
.. 545      P 0543      1 END;
```

```
.. 545      P 0543      1 END;
```

4F	49	5F	47	4F	4C	20	6B	63	61	6C	20	75	6F	59	00000	P.AAA:	.ASCII	\You lack LOG_IO privilege.\
20	3D	3E	20	65	62	20	74	73	75	6D	20	4E	42	56	0000F			
6F	6E	20	73	69	20	65	6C	69	66	20	74	6F	6F	42	0001A	P.AAB:	.ASCII	\VBN must be >= 1.\
61	20	44	41	45	52	20	6B	63	61	6C	20	75	6F	59	0002B	P.AAC:	.ASCII	\Boot file is not contiguous.\
63	63	61	20	45	54	49	52	57	20	72	6F	2F	64	6E	0003A	P.AAD:	.ASCII	\You lack READ and/or WRITE access to TAR\
49	44	20	20	2E	45	43	49	56	45	44	20	54	45	47	00047		.ASCII	\GET DEVICE. DISMOUNT and reMOUNT it.\
4F	4D	65	72	20	64	6E	61	20	54	4E	55	4F	4D	53	00056			
64	20	6D	65	74	73	79	73	20	74	65	67	72	61	5D	00065	P.AAE:	.ASCII	\J\
74	6F	6F	62	20	64	6E	61	28	20	65	63	69	76	65	0007E	P.AAF:	.ASCII	\Target system device (and boot file if n\
72	64	64	61	20	64	61	6F	6C	20	72	65	74	6E	45	0008D		.ASCII	\ot VMB.EXE): \
20	79	72	61	6D	69	72	70	20	66	6F	20	73	73	65	00094	P.AAG:	.ASCII	\Enter load address of primary bootstrap \
74	6C	75	61	66	65	64	28	20	58	45	48	20	6E	69	000A4		.ASCII	\in HEX (default is 200): \
6F	62	20	66	6F	20	4E	42	56	20	72	65	74	6E	45	000BD	P.AAH:	.ASCII	\Enter VBN of boot file code (default is \
64	28	20	65	64	6F	63	20	65	6C	69	66	20	74	6F	000CA			
45	58	45	2E	42	4D	56	5D	45	58	45	53	59	53	5B	000D9	P.AAI:	.ASCII	\1) : \
45	58	45	2E	3A	4D	4B	43	4F	4C	42	54	4F	4F	42	000E8	P.AAJ:	.ASCII	\[SYSEXE]VMB.EXE\
															000F2	P.AAK:	.ASCII	\BOOTBLOCK\
															00101		.ASCII	\SYSS\$SYSTEM:.EXE\
															0010B			
															0011A			
															00129			
															00133			
															00138			
															00147			
															00150			

```
.PSECT $PLITS$,NOWRT,NOEXE,2
```

```
00# 00000 PRIBOO_DESCRIP:
02 00003 .BYTE 0[3]
00# 00004 .BYTE 2
00# 00008 LOADADR_DESCRIP:
02 0000B .BYTE 0[3]
00# 0000C .BYTE 2
00# 00010 PROMPT_DESCRIP:
01 00013 .BYTE 0[3]
00014 .BLKB 1
00# 00018 VBN_DESCRIP:
02 0001B .BYTE 0[3]
00# 0001C .BYTE 2
00020 PRIBOO_FAB:
00070 BOOTBL_FAB:
000C0 PRIBOO_FILNAM:
001BF .BLKB 80
001C0 PRIBOO_EXP_NAME:
002BF .BLKB 80
02 002C0 PRIBOO_NAM_BLK:
60 002C1 .BLKB 255
FF 002C2 .BLKB 1
00 002C3 .BLKB 1
00000000' 002C4 .BLKB 255
00 002C8 .BLKB 1
00 002C9 .BLKB 1
FF 002CA .BLKB 1
00 002CB .BLKB 1
00000000' 002CC .BLKB 255
00000000 002D0 .BLKB 1
0000# 002D4 .BLKB 1
0000# 002E4 .BLKB 1
0000# 002EA .BLKB 1
00000000 002F0 .BLKB 1
00000000 002F4 .BLKB 1
00 002F8 .BLKB 1
00 002F9 .BLKB 1
00 002FA .BLKB 1
00 002FB .BLKB 1
00 002FC .BLKB 1
00 002FD .BLKB 1
00# 002FE .BLKB 1
00000000 00300 .BLKB 1
00000000 00304 .BLKB 1
00000000 00308 .BLKB 1
00000000 0030C .BLKB 1
00000000 00310 .BLKB 1
00000000 00314 .BLKB 1
```


M 4
15-Sep-1984 23:36:40
14-Sep-1984 11:58:06VAX-11 Bliss-32 V4.0-742
[BOOTS.SRC]WRITEBOOT.B32;1Page 13
(3)

```
00000000# 00318 .LONG 0[2]
      1D 00320 PRIBOO_XABFHC:
      2C 00321 .BYTE 29
      0000 00322 .BYTE 44
      00000000 00324 .WORD 0
      00000000# 00328 .LONG 0
      0034C PRIVILEGE_MASK:
      .BLKB 8
0204 0008 00354 GETJPI_ITEMLIST:
      .WORD 8, 516
00000000 00000000' 00358 .ADDRESS PRIVILEGE_MASK
00000000 00000000 0035C .LONG 0, 0
      00364 IO_STAT_BLOCK:
      .BLKB 8
      0036C TWO_BLOCK_BUF:
      .BLKB 1024
      0076C BOOTDEV_DESCRIP:
      .BLKB 8
      00774 BOOTDEV_CHAN:
      .BLKB 2
      00776 .BLKB 2
      00778 LOAD_ADR:
      .BLKB 4
00000000# 0077C DEVCHAR_BUFF:
      .LONG 0[3]
0000000C 00788 DEVCHAR_DESCRIP:
      .LONG 12
00000000' 0078C .ADDRESS DEVCHAR_BUFF
      00# 00790 FILNAM_DESCRIP:
      .BYTE 0[8]
00000000 00798 FILSPEC_DESCRIP:
      .LONG 0
00000000' 0079C .ADDRESS PRIBOO_EXP_NAME
      007A0 VBN: .BLKB 4
      007A4 STAT_BLOCK:
      .BLKB 8
      00 007AC YES_NO_BUF:
      .BYTE 0
      007AD .BLKB 3
00000001 007B0 YES_NO_DESCRIP:
      .LONG 1
00000000' 007B4 .ADDRESS YES_NO_BUF
0000001A 007B8 LOGIO_DESCRIP:
      .LONG 26
00000000' 007BC .ADDRESS LOGIO_MSG
00000011 007C0 VBN_BNDS_DESCRIP:
      .LONG 17
00000000' 007C4 .ADDRESS VBN_BNDS_MSG
0000004D 007C8 REMOUNT_DESCRIP:
      .LONG 77
00000000' 007CC .ADDRESS REMOUNT_MSG
0000001C 007D0 NOTCONTIG_DESCRIP:
      .LONG 28
00000000' 007D4 .ADDRESS NOTCONTIG_MSG
00000001 007D8 BRACKET_DESCRIP:
      .LONG 1
```

00000000' 007DC

.ADDRESS ASCII_BRACKET

```
BLOCK_BUFFER= TWO_BLOCK_BUF
LOGIO_MSG= P.AAA
VBN_BNDS_MSG= P.AAB
NOTCONTIG_MSG= P.AAC
REMOUNT_MSG= P.AAD
ASCII_BRACKET= P.AAE
PROMPT_BUFFER= P.AAF
PROMPT2_BUFFER= P.AAG
PROMPT3_BUFFER= P.AAH
PRIBOO_DEF_NAME= P.AAI
SRMS_PTR= PRIBOO_FAB
SRMS_PTR= BOOTBL_FAB
.EXTRN OTSSCVT_TZ_L, LIB$INDEX
.EXTRN LIB$PUT_OUTPUT, LIB$FREE1_DD
.EXTRN RTF$TARGET_DEV, RTF$OPENFICE
.EXTRN LIB$GET_INPUT, SYSS$GETJPI
.EXTRN SYSS$PARSE, SYSS$GETDEV
.EXTRN SYSS$OPEN, SYSS$CLOSE
.EXTRN SYSS$QIOW, SYSS$ASSIGN
```

.PSECT \$CODE\$,NOWRT,2

OFFC 00000 WRITE_BOOT:

5B	00000000G	00	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	0240
5A	00000000G	00	9E	00009	MOVAB	SYSS\$CLOSE, R11	
59	00000000G	00	9E	00010	MOVAB	LIB\$GET_INPUT, R10	
58	00000000G	00	9E	00017	MOVAB	LIB\$FREE1_DD, R9	
57	0000'	CF	9E	0001E	MOVAB	LIB\$PUT_OUTPUT, R8	
		7E	7C	00023	MOVAB	PROMPT_DESCRIP, R7	
	0354	C7	9F	00025	CLRQ	-(SP)	0295
	0344	C7	9F	00029	PUSHAB	IO_STAT_BLOCK	
		7E	7C	0002D	PUSHAB	GETJPI_ITEMLIST	
		03	DD	0002F	CLRQ	-(SP)	
00000000G	00	07	FB	00031	PUSHL	#3	
56		50	D0	00038	CALLS	#7, SYSS\$GETJPI	
38		56	E9	0003B	MOVL	R0, STATUS	
	033C	C7	95	0003E	BLBC	STATUS, 3\$	
		0B	19	00042	TSTB	PRIVILEGE_MASK	0297
	07A8	C7	9F	00044	BLSS	1\$	
68		01	FB	00048	PUSHAB	LOGIO_DESCRIP	0299
50		24	D0	0004B	CALLS	#1, LIB\$PUT_OUTPUT	
		04	00	0004E	MOVL	#36, R0	0300
67		35	B0	0004F	RET		
04	A7	CF	9E	00052	1\$: MOVW	#53, PROMPT_DESCRIP	0308
	F0	A7	B5	00058	MOVAB	PROMPT_BUFFER, PROMPT_DESCRIP+4	0309
		06	13	0005B	TSTW	PRIBOO_DESCRIP	0311
	F0	A7	9F	0005D	BEQL	2\$	
69		01	FB	00060	PUSHAB	PRIBOO_DESCRIP	0312
	F0	A7	B5	00063	CALLS	#1, LIB\$FREE1_DD	
		11	12	00066	TSTW	PRIBOO_DESCRIP	0314
		57	DD	00068	BNEQ	4\$	
	F0	A7	9F	0006A	PUSHL	R7	0317
6A		02	FB	0006D	PUSHAB	PRIBOO_DESCRIP	
56		50	D0	00070	CALLS	#2, LIB\$GET_INPUT	
					MOVL	R0, STATUS	

0050	8F	E7 00	ED	56	E8	00073	BLBS	STATUS, 2\$	0318
			51	F0	02AD	31	BRW	24\$	0319
			50		A7	3C	MOVZWL	PRIBOO_DESCRIP, R1	0327
					01	CE	MNEGL	#1, COUNT	0331
					15	11	BRB	6\$	
	61	8F	F4	B740	91	00082	CMPB	@PRIBOO_DESCRIP+4[COUNT], #97	0332
					0D	1F	BLSSU	6\$	
	7A	8F	F4	B740	91	0008A	CMPB	@PRIBOO_DESCRIP+4[COUNT], #122	
					05	1A	BGTRU	6\$	
	F4	B740			20	82	SUBB2	#32, @PRIBOO_DESCRIP+4[COUNT]	0333
					51	F2	AOBLS	R1, COUNT, 5\$	0327
					00	2C	MOVC5	#0, (SP), #0, #80, \$RMS_PTR	0354
					A7	000A2			
	10	A7	5003	8F	B0	000A4	MOVW	#20483, \$RMS_PTR	
	14	A7	01000000	8F	D0	000AA	MOVL	#16777216, \$RMS_PTR+4	
	26	A7		02	90	000B2	MOVB	#2, \$RMS_PTR+22	
	2F	A7		02	90	000B6	MOVB	#2, \$RMS_PTR+31	
	34	A7	0310	C7	9E	000BA	MOVAB	PRIBOO_XABFHC, \$RMS_PTR+36	
	38	A7	02B0	C7	9E	000C0	MOVAB	PRIBOO_NAM_BLK, \$RMS_PTR+40	
	3C	A7	F4	A7	D0	000C6	MOVL	PRIBOO_DESCRIP+4, \$RMS_PTR+44	
	40	A7	0000	CF	9E	000CB	MOVAB	PRIBOO_DEF_NAME, \$RMS_PTR+48	
	44	A7	F0	A7	90	000D1	MOVB	PRIBOO_DESCRIP, \$RMS_PTR+52	
	45	A7		0F	90	000D6	MOVB	#15, \$RMS_PTR+53	
				A7	9F	000DA	PUSHAB	PRIBOO_FAB	0355
	00000000G	00		01	FB	000DD	CALLS	#1, SYS\$PARSE	
		56		50	D0	000E4	MOVL	R0, STATUS	
		7D		56	E9	000E7	BLBC	STATUS, 7\$	
	075C	C7	02C4	C7	9B	000EA	MOVZBW	PRIBOO_NAM_BLK+20, BOOTDEV_DESCRIP	0358
	0760	C7	02C5	C7	9E	000F1	MOVAB	PRIBOO_NAM_BLK+21, BOOTDEV_DESCRIP+4	0359
				7E	7C	000F8	CLRQ	-(SP)	0363
			0778	C7	9F	000FA	PUSHAB	DEVCHAR_DESCRIP	
				7E	D4	000FE	CLRL	-(SP)	
			075C	C7	9F	00100	PUSHAB	BOOTDEV_DESCRIP	
	00000000G	00		05	FB	00104	CALLS	#5, SYS\$GETDEV	
		56		50	D0	0010B	MOVL	R0, STATUS	
		56		56	E9	0010E	BLBC	STATUS, 7\$	
		54	076F	C7	E9	00111	BLBC	DEVCHAR_BUFF+3, 8\$	0380
	0788	C7	02BB	C7	9B	00116	MOVZBW	PRIBOO_NAM_BLK+11, FILSPEC_DESCRIP	0382
			07C8	C7	9F	0011D	PUSHAB	BRACKET_DESCRIP	0383
			0788	C7	9F	00121	PUSHAB	FILSPEC_DESCRIP	
	00000000G	00		02	FB	00125	CALLS	#2, LIB\$INDEX	
	0784	C7	078C	D740	9E	0012C	MOVAB	@FILSPEC_DESCRIP+4[INDEX], FILNAM_DESCRIP+4	0384
0780	C7	0788	C7	50	A3	00134	SUBW3	INDEX, FILSPEC_DESCRIP, FILNAM_DESCRIP	0385
			075C	C7	9F	0013C	PUSHAB	BOOTDEV_DESCRIP	0387
	00000000G	00		01	FB	00140	CALLS	#1, RTF\$TARGET_DEV	
			0794	C7	9F	00147	PUSHAB	STAT_BLOCK	0389
			035C	C7	9F	0014B	PUSHAB	TWO_BLOCK_BUF	
			0780	C7	9F	0014F	PUSHAB	FILNAM_DESCRIP	
	00000000G	00		03	FB	00153	CALLS	#3, RTF\$OPENFILE	
		56		50	D0	0015A	MOVL	R0, STATUS	
		53		56	E8	0015D	BLBS	STATUS, 12\$	
			07B8	C7	9F	00160	PUSHAB	REMOUNT_DESCRIP	0393
		68		01	FB	00164	CALLS	#1, LIB\$PUT_OUTPUT	
				31	00167	7\$:	BRW	24\$	0394
			10	A7	9F	0016A	PUSHAB	PRIBOO_FAB	0398
	00000000G	00		01	FB	0016D	CALLS	#1, SYS\$OPEN	
		56		50	D0	00174	MOVL	R0, STATUS	

0794	ED	0338	56	E9	00177	BLBC	STATUS, 7\$	0401
	C7		C7	D0	0017A	MOVL	PRIBOO_XABFHC+40, STAT_BLOCK	0402
		07C0	13	12	00181	BNEQ	9\$	0404
	68		C7	9F	00183	PUSHAB	NOTCONTIG DESCRIP	
		10	01	FB	00187	CALLS	#1, LIB\$PUT_OUTPUT	0405
	6B		A7	9F	0018A	PUSHAB	PRIBOO_FAB	
	50	02AC	01	FB	0018D	CALLS	#1, SYSSCLOSE	0406
			8F	3C	00190	MOVZWL	#684, R0	
		0324		04	00195	RET		0408
			C7	B5	00196	TSTW	PRIBOO_XABFHC+20	
			09	13	0019A	BEQL	10\$	0409
0798	C7	0320	C7	D0	0019C	MOVL	PRIBOO_XABFHC+16, STAT_BLOCK+4	
			08	11	001A3	BRB	11\$	0410
0798	C7	0320	C7	01	C3	SUBL3	#1, PRIBOO_XABFHC+16, STAT_BLOCK+4	0411
				A7	9F	PUSHAB	PRIBOO_FAB	
	6B	10	01	FB	001B0	CALLS	#1, SYSSCLOSE	0419
	67		2D	B0	001B3	MOVW	#45, PROMPT_DESCRIP	0420
	04	0000'	CF	9E	001B6	MOVAB	PROMPT3_BUFFER, PROMPT_DESCRIP+4	0422
			56	D4	001BC	CLRL	STATUS	0423
	4C		56	E8	001BE	BLBS	STATUS, 19\$	0426
		08	A7	B5	001C1	TSTW	VBN_DESCRIP	
			06	13	001C4	BEQL	15\$	0427
		08	A7	9F	001C6	PUSHAB	VBN_DESCRIP	
	69		01	FB	001C9	CALLS	#1, LIB\$SFREE1_DD	0429
			57	DD	001CC	PUSHL	R7	
		08	A7	9F	001CE	PUSHAB	VBN_DESCRIP	
	6A		02	FB	001D1	CALLS	#2, LIB\$GET_INPUT	
	56		50	D0	001D4	MOVL	R0, STATUS	
	30		56	E9	001D7	BLBC	STATUS, 18\$	0432
		08	A7	B5	001DA	TSTW	VBN_DESCRIP	
			13	13	001DD	BEQL	16\$	0433
		0790	C7	9F	001DF	PUSHAB	VBN	
		08	A7	9F	001E3	PUSHAB	VBN_DESCRIP	
00000000G	00		02	FB	001E6	CALLS	#2, OTSSCVT_TZ_L	
	56		50	D0	001ED	MOVL	R0, STATUS	
			05	11	001F0	BRB	17\$	
0790	C7		01	D0	001F2	MOVL	#1, VBN	0434
		0790	C7	D5	001F7	TSTL	VBN	0436
			C1	14	001FB	BGTR	14\$	
		07B0	C7	9F	001FD	PUSHAB	VBN_BNDS_DESCRIP	0439
	68		01	FB	00201	CALLS	#1, LIB\$PUT_OUTPUT	
	56		50	D0	00204	MOVL	R0, STATUS	
	B2		56	E8	00207	BLBS	STATUS, 13\$	
			0119	31	0020A	BRW	24\$	0440
50	0794	C7	C7	C1	0020D	ADDL3	VBN, STAT_BLOCK, R0	0445
	0794	FF	A0	9E	00215	MOVAB	-1(R0), STAT_BLOCK	
0050	8F	00	00	2C	0021B	MOVCS	#0, (SP), #0, #80, \$RMS_PTR	0457
		60	A7		00222			
	60	A7	8F	B0	00224	MOVW	#20483, \$RMS_PTR	
	64	A7	8F	D0	0022A	MOVL	#131072, \$RMS_PTR+4	
	76	A7	20	90	00232	MOVB	#32, \$RMS_PTR+22	
	7F	A7	02	90	00236	MOVB	#2, \$RMS_PTR+31	
008C	C7	0000'	CF	9E	0023A	MOVAB	P.AAJ, \$RMS_PTR+44	
0090	C7	0000'	CF	9E	00241	MOVAB	P.AAK, \$RMS_PTR+48	
0094	C7	0F09	8F	B0	00248	MOVW	#3849, \$RMS_PTR+52	
		60	A7	9F	0024F	PUSHAB	BOOTBL_FAB	0458
00000000G	00		01	FB	00252	CALLS	#1, SYSSOPEN	

	56		50	D0	00259	MOVL	R0, STATUS		
	AB		56	E9	0025C	BLBC	STATUS, 18\$		
	7E		7E	7C	0025F	CLRQ	-(SP)	0472	
	7E		01	7D	00261	MOVQ	#1, -(SP)		
	7E	0200	8F	3C	00264	MOVZWL	#512, -(SP)		
		035C	C7	9F	00269	PUSHAB	BLOCK_BUFFER		
			7E	7C	0026D	CLRQ	-(SP)		
	7E		31	7D	0026F	MOVQ	#49, -(SP)		
		6C	A7	DD	00272	PUSHL	BOOTBL_FAB+12		
			7E	D4	00275	CLRL	-(SP)		
00000000G	00		0C	FB	00277	CALLS	#12, SYS\$QIOW		
	56		50	D0	0027E	MOVL	R0, STATUS		
	86		56	E9	00281	BLBC	STATUS, 18\$		
	67	41	8F	9B	00284	MOVZBW	#65, PROMPT_DESCRIP	0480	
04	A7	0000	CF	9E	00288	MOVAB	PROMPT2_BUFFER, PROMPT_DESCRIP+4	0481	
			56	D4	0028E	CLRL	STATUS	0483	
	3A		56	E8	00290	BLBS	STATUS, 23\$	0485	
		F8	A7	B5	00293	TSTW	LOADADR_DESCRIP	0488	
			06	13	00296	BEQL	21\$		
		F8	A7	9F	00298	PUSHAB	LOADADR_DESCRIP	0489	
	69		01	FB	0029B	CALLS	#1, LIB\$SFREE1_DD		
			57	DD	0029E	PUSHL	R7	0491	
		F8	A7	9F	002A0	PUSHAB	LOADADR_DESCRIP		
	6A		02	FB	002A3	CALLS	#2, LIB\$GET_INPUT		
	56		50	D0	002A6	MOVL	R0, STATUS		
	7A		56	E9	002A9	BLBC	STATUS, 24\$	0492	
		F8	A7	B5	002AC	TSTW	LOADADR_DESCRIP	0494	
			13	13	002AF	BEQL	22\$		
		0768	C7	9F	002B1	PUSHAB	LOAD_ADR	0495	
		F8	A7	9F	002B5	PUSHAB	LOADADR_DESCRIP		
00000000G	00		02	FB	002B8	CALLS	#2, OT\$SCVT_TZ_L		
	56		50	D0	002BF	MOVL	R0, STATUS		
			CC	11	002C2	BRB	20\$		
	0768	C7	8F	3C	002C4	MOVZWL	#512, LOAD_ADR	0496	
			C3	11	002CB	BRB	20\$	0485	
	035C	C7	C7	D0	002CD	MOVL	STAT_BLOCK+4, BLOCK_BUFFER	0504	
	0360	C7	C7	B0	002D4	MOVW	STAT_BLOCK+2, BLOCK_BUFFER+4	0505	
	0362	C7	C7	B0	002DB	MOVW	STAT_BLOCK, BLOCK_BUFFER+6	0506	
	0364	C7	C7	D0	002E2	MOVL	LOAD_ADR, BLOCK_BUFFER+8	0507	
			7E	7C	002E9	CLRQ	-(SP)	0517	
		0764	C7	9F	002EB	PUSHAB	BOOTDEV_CHAN		
		075C	C7	9F	002EF	PUSHAB	BOOTDEV_DESCRIP		
00000000G	00		04	FB	002F3	CALLS	#4, SYS\$ASSIGN		
	56		50	D0	002FA	MOVL	R0, STATUS		
	26		56	E9	002FD	BLBC	STATUS, 24\$		
			7E	7C	00300	CLRQ	-(SP)	0529	
			7E	7C	00302	CLRQ	-(SP)		
	7E	0200	8F	3C	00304	MOVZWL	#512, -(SP)		
		035C	C7	9F	00309	PUSHAB	BLOCK_BUFFER		
			7E	7C	0030D	CLRQ	-(SP)		
	7E		20	7D	0030F	MOVQ	#32, -(SP)		
	7E	0764	C7	3C	00312	MOVZWL	BOOTDEV_CHAN, -(SP)		
			7E	D4	00317	CLRL	-(SP)		
00000000G	00		0C	FB	00319	CALLS	#12, SYS\$QIOW		
	56		50	D0	00320	MOVL	R0, STATUS		
	04		56	E8	00323	BLBS	STATUS, 25\$		
	50		56	D0	00326	MOVL	STATUS, R0	0530	

WRITEBOOT
V04-000

E 5
15-Sep-1984 23:36:40
14-Sep-1984 11:58:06

VAX-11 BLISS-32 V4.0-742
[BOOTS.SRC]WRITEBOOT.B32;1

Page 18
(3)

```

        6B      60      A7      04 00329      RET
        50      01      9F 0032A 25$:  PUSHAB  BOOTBL FAB
        01      01      FB 0032D      CALLS  #1, SYSSCLOSE
        04      01      D0 00330      MOVL   #1, R0
        04      04      04 00333      RET

```

:
: 0536
:
: 0542
: 0543

; Routine Size: 820 bytes, Routine Base: \$CODE\$ + 0000

: 546 0544 1 END
: 547 0545 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	2016	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	351	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	820	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	82	0	1000	00:01.9

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:WRITEBOOT/OBJ=OBJ\$:WRITEBOOT MSRC\$:WRITEBOOT/UPDATE=(ENH\$:WRITEBOOT)

; Size: 820 code + 2367 data bytes
; Run Time: 00:20.7
; Elapsed Time: 00:25.9
; Lines/CPU Min: 1577
; Lexemes/CPU-Min: 29927
; Memory Used: 276 pages
; Compilation Complete

0042 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY